

Portable Program Review

Vol. 1, No. 7

January, 1986

GET THE QBUG

Let's face it, debugging machine-language programs is difficult and frustrating. Thanks to Jay Holovacs of Warren, New Jersey, Qbug can help.

The Qbug listing can be downloaded from CompuServe's Model 100 SIG (go ml00sig). It's in data library 4 under the name QBUG.BA.

The author can be contacted on CompuServe at user id 74756,419. -Ed.

The Qbug is intended to simplify debugging of machine-language programs on the Model 100 by providing interactive user information and control of the CPU registers, stack, memory locations, break and execution points.

Qbug accomplishes this by accessing your code through a ML link rather than direct call. The values selected by the user are transferred to the appropriate registers and then the routine is called. When returning, the linkage

saves stack and register values for interpretation through Qbug.

In general, machine language programs should be debugged much more meticulously than BASIC. Below are some suggestions:

- * Always work from a printed copy of assembler output listing. Consult it meticulously; don't rely on memory.
- * Execute small stretches of code at a time setting breakpoints at key areas (always set a breakpoint at a conditional branch the first time through with each set of trial values. Use the Registers command of Qbug to check condition of the flags). I find it helpful to repeatedly use the same execution entry point and data, moving the breakpoint further down the line as the segments are verified; then choosing new entry values.
- * It may be necessary to assemble a test version of your routine at a location different from your final working version to keep from interfering with Qbug. Proper use of labels in your source code will allow the reassembly at the running location to proceed without incident.
- * When testing any routines which access a system function or value which can affect BASIC (a hook into the background task, for example) always test with dummy memory locations referenced instead. You can easily insert and read values in these areas without danger of crashing the

IN THIS ISSUE:

GET THE QBUG.....1

ADDRESS NOT UNKNOWN.....6

ALMOST PERFECTION.....7

debugger (or all your files). Only after the routine is verified to perform exactly as expected should the actual addresses be assembled in place.

- * Never execute a stretch of new code without having tape copies of everything in RAM. Proper use of the debugger can greatly reduce chances of a runaway program, but occasional mistakes are inevitable.

Using Qbug

When the program is run, the maximum user address (F53C in current version 1.2.2) is calculated and displayed on the title screen.

Before running the program, load the code to be tested and set HIMEM. Be sure that you don't overwrite the maximum user address. Qbug will check HIMEM and halt if it doesn't appear to be correct.

After the title screen there is a brief pause as the support routine is loaded; then the menu is presented. All control of your program is through the menu options. After executing each option, Qbug returns to the menu; all options (except Quit) can be repeatedly called, reviewed and changed before executing the code.

Simply press the letter key for each command from menu. Every option can be aborted after selection in case of an error.

NOTE: Always leave CAPS LOCK on and enter all values in hexadecimal when using Qbug. The backspace key is used to make corrections when entering values in hexadecimal.

Commands and Options

- * Registers -- Allows you to preset and examine values in registers A-L and Flag before and after executing a section of code. As the value in each register is presented hit Enter to leave it unchanged or [hex value] Enter to set a value.
- * Memory -- when prompted enter a starting address; now the memory locations, values, and characters are displayed and can be changed the same way the registers were. While viewing memory values, you use the cursor control keys (UP for next location, DOWN for previous location) to move around or press Enter to return to menu. To look at an area of memory far removed, return to menu and re-select the Memory option.
- * Breakpoint -- Use this to set the stop point (address) for your test run. The execution stops just before the addresses you specify. If a breakpoint is already set, it is displayed upon entry and can be left unchanged by simply hitting Enter. If another breakpoint is selected, the first one is automatically cleared. Breakpoints are cleared and must be reset after Executing code. Selecting a breakpoint replaces 3 bytes of your code with a jump into the cleanup part of the ML support. When a breakpoint is cleared your original values are automatically placed back. DO NOT exit your code in any way except through a breakpoint; the cleanup routines are essential for the return to Qbug.
- * Execute -- Allows the execution of a section of code starting at the address specified and ending at the breakpoint (Execute will automatically abort if no breakpoint is set). When entering, you are prompted for a starting address; hitting Enter will abort to menu. After the starting address you may opt to place up to 5 values on the stack; these are 2 byte entries which will be on top of the stack when the actual execution begins. Hitting Enter or "0" will bypass this option.
- * After execution, breakpoints are cleared and you are returned to

the main menu.

* Stack -- Allows inspection of the top 5 items "user stack" -- that is, items remaining on the stack at the end of the most recent execution which are not part of the BASIC or Qbug operations. While the stack can be inspected from this option, it can only be loaded as part of the Execute operation.

* Quit -- Always leave the program through the Quit option; this clears any pending breakpoints to leave your code intact.

Note that Qbug will not accept breakpoints, entry points, or changes to memory addresses outside your program area.

Tech Notes

When selecting breakpoint locations, bear in mind that you cannot RETURN up through more layers on the stack than you have CALLED unless the appropriate return addresses have been placed there ahead of time. Qbug uses the warm boot address as its flag to determine what part of the stack belongs to the code being evaluated; if you accidentally RETURN through this a warm boot will be executed -- saving you from a system crash. This isn't bulletproof, however. If you use a lot of PUSHes and POPs you could conceivably go through this protection.

It is possible for an error in the user program to overwrite part of Qbug, other files or system utilities in RAM. Always keep tape copies of everything in RAM when exercising a program. If you start having trouble with Qbug, perform a cold boot and reload. For example, part of BASIC's CALL statement executes out of RAM -- if this code is overwritten, the statement will not execute properly.

To allow the support to be reassembled for different locations, the ML entry points are

handled through variables (see REMarks) set at the beginning of the program. If necessary the source code supplied can be reassembled at different addresses.

Listing of QBUG.BA, the machine-language debugger.

```
5 V$ = "1.2.2 CIS" 'QBUG DEBUGGER REV 8/30/85
10 REM BY JAY HOLOVACS 95 KING GEORGE RD.
   WARREN NJ 07060
15 REM INCORPORATES DBUG-ML ROUTINE V1.1--WHEN
   MAKING ML SUPPORT CHANGES VERIFY LINES 20,
   30, 200, 205; variables PC!, HM, SK, ST
20 H$ = "0123456789ABCDEF"
   :DEFINT R, P, Q
   :DEFSNG D, B, H
   :DIM B1(3)
   :PC! = 62798
   :HM = 62780
25 REM PC! = BASE ADDRESS OF REGISTER TRANSFER
   AREA; HM = BEGINNING OF ML ROUTINE-2(TO
   ALLOW FOR BREAKPOINT)
30 SK = 62782
   :ST = 62808 'STACK COUNT POINTER; START
   ROUTINE; EXIT ROUTINE
35 GOSUB 580
   :IF (CL AND 32) = 32 THEN BEEP
   :PRINT "> Please depress CAPS LOCK"
   :LINE INPUT " [enter] to continue"; R$
   :GOTO 35
40 DC = HM
   :GOSUB 360
   :CLS
   :PRINT "* * QBUG DEBUGGER PROGRAM ver ";
   V$
45 PRINT " by Jay Holovacs"
   :PRINT ">Load your program and adjust HIMEM"
50 PRINT " BEFORE running QBUG."
   :PRINT ">Don't use addresses above "; H$; "
   ("; STR$(HM); ");"
55 PRINT ">All NUMERIC DATA is in HEXADECIMAL"
60 PRINT
   :LINE INPUT " [enter] to continue"; R$
65 IF HIMEM < HM! THEN 75 ELSE BEEP
   :CLS
   :PRINT "M/L Program not present or not
   located"
70 PRINT "at proper address range (below ";
   H$; ")"
   :PRINT "--please reload and adjust HIMEM"
   :END
75 GOSUB 585 'load ML support above user's
   program
80 CLS
   :PRINT "R: load/examine REGISTERS ";
   CHR$(27); "p"; "COMMANDS:"; CHR$(27); "q"
85 PRINT "M: load/examine MEMORY"
   :PRINT "B: set/examine BREAKPOINT"
   :PRINT "E: EXECUTE program"
90 PRINT "S: Inspect user STACK"
   :PRINT "Q: QUIT debugger"
   :PRINT "H: Hex-to-Decimal conversion"
   :PRINT "D: Decimal-to-Hex conversion";
95 R$ = INKEY$
   :IF R$ = "" THEN 95
100 R = INSTR("RMBEQSHD", R$)
   :IF R = 0 THEN 95
```



```

105 ON R GOSUB 110, 215, 160, 280, 415, 485,
535, 560
:GOTO 80
110 REM LOAD REGISTERS
115 RESTORE 155
:CLS
:PRINT "*" * LOAD REGISTERS* "*"
:PRINT "Enter blank for no change"
:FOR N = 1 TO 8
:READ RG$, R
120 DC = PEEK(PC! + R)
:GOSUB 360
:PRINT RG$; "-"; RIGHT$(H$, 2); " ";
:GOSUB 445
:GOSUB 390
:IF H$ = "" THEN 135 'skip poke if no change
125 IF DC > 255 OR DC < 0 THEN BEEP
:PRINT " ERROR--must be 00-FF"
:GOTO 120
130 POKE PC! + R, DC
135 PRINT
140 IF N <> 7 THEN 150 ELSE PRINT @183,
"Status Flags"
:PRINT @220, "S Z * * * PV * C"
145 PRINT @260, "8 4 2 1 8 4 2 1"
:LINE (115, 30) - (225, 60), 1, B
150 NEXT N
:RETURN
155 DATA A REG, 2, B REG, 3, C REG, 4, D REG,
5, E REG, 6, H REG, 8, L REG, 7, FLAGS, 9
160 REM SET UP BREAKPOINT
165 RESTORE 205
:CLS
:PRINT "*" *BREAKPOINT* "*"
170 IF B1(3) <> 0 THEN DC = B1(3)
:GOSUB 360
:PRINT "CURRENT BREAKPOINT = "; H$
:PRINT "(Set a new breakpoint to clear this
one)"
175 PRINT "SET BREAKPOINT (enter blank to
return)"
:PRINT "-->";
:GOSUB 445
:IF H$ = "" THEN RETURN
180 GOSUB 390
:IF DC < HIMEM OR DC >= HM THEN BEEP
:PRINT " Error--outside of program area"
:GOTO 175
185 GOSUB 435 'CLEAR OLD BREAKPOINT
190 B1(3) = DC
:FOR P = 0 TO 2
:B1(P) = PEEK(B1(3) + P)
195 READ N
:POKE B1(3) + P, N
:NEXT P 'INSERT JUMP INSTRUCTION
200 REM JMP, 94, F5 = JUMP 62853 = EXIT
ROUTINE
205 DATA 195, 148, 245
210 RETURN
215 REM LOAD MEMORY WITH VALUES
220 CLS
:PRINT "*" *LOAD MEMORY* "*"
:PRINT "To CHANGE a value; type the
desired"
225 PRINT "value and proceed. To leave it UN-"
:PRINT "CHANGED, move on without typing a
value.";
:PRINT "*" *CURSOR KEYS:"
230 PRINT CHR$(152); " = NEXT ADDRESS ";
CHR$(153); " = PREV. ADDRESS"
:PRINT "[enter] = RETURN TO MENU"

```

```

235 PRINT "LOCATION?->";
:GOSUB 445
:PRINT
:GOSUB 390
:ML$ = H$
:ML = DC
240 MV = PEEK(ML)
:DC = MV
:GOSUB 360
:PRINT ML$; "->"; RIGHT$(H$, 2); "-";
:IF MV > 32 THEN PRINT CHR$(MV); " "; ELSE
PRINT " ";
245 GOSUB 445
:IF H$ = "" THEN 255 ELSE IF LEN(H$) < 3
AND ML > HIMEM AND ML < HM THEN GOSUB 390
:POKE ML, DC
:GOTO 255
250 IF LEN(H$) > 2 THEN PRINT " ERROR--val
must be 00-FF" ELSE PRINT " Address is in
protected"
:PRINT "area"
:GOTO 240
255 PRINT
260 IF ASC(R$) = 13 THEN RETURN
265 IF ASC(R$) = 31 THEN ML = ML - 1
:GOTO 275
270 IF ASC(R$) = 30 THEN ML = ML + 1
275 DC = ML
:GOSUB 360
:ML$ = H$
:GOTO 240
280 REM EXECUTE A PROGRAM
285 CLS
:PRINT "*" *EXECUTE PROGRAM* "*"
290 IF B1(3) = 0 THEN BEEP
:PRINT " ERROR--No breakpoint set"
:LINE INPUT "[enter] to Abort.."; R$
:RETURN
295 PRINT "Enter starting address (blank to
abort)"
:PRINT "-->";
:GOSUB 445
:IF H$ = "" THEN RETURN
300 GOSUB 390
:IF DC <= HIMEM OR DC >= HM THEN BEEP
:PRINT " ERROR--Out of your program area"
:GOTO 295
305 HB = INT(DC / 256)
:POKE PC! + 1, HB
:POKE PC!, DC - HB * 256 'LOAD STARTING
ADDRESS
310 CLS
315 N = 0
:PRINT @0, "Number of items (0 to 5) to
place on "
:PRINT @40, "stack--Most Accessible (top)
first"
320 PRINT "or [enter] to skip";
:INPUT N
:IF N > 5 THEN BEEP
:GOTO 315
325 N = N * 2
:POKE SK, N
:IF N = 0 THEN 345 'LOOP BELOW WILL ITERATE
ONCE EVEN IF N = 0
330 FOR QQ = 2 TO N STEP 2
:PRINT "-->";
:GOSUB 445
:PRINT
:H$ = RIGHT$("0000" + H$, 4)
:HB$ = LEFT$(H$, 2)

```



```

335 LB$ = RIGHT$(H$, 2)
: H$ = LB$
: GOSUB 390
: POKE SK + QQ, DC 'POKE LO BYTE VALUE
340 H$ = HB$
: GOSUB 390
: POKE SK + QQ + 1, DC
: NEXT QQ 'POKE HIGH BYTE VALUE
345 CALL ST 'ADDRESS OF START LABEL
350 GOSUB 435 'CLEAR BREAKPOINT
355 RETURN
360 REM DEC-HEX CONVERSION
365 REM ENTER
: DC = DECIMAL; EXIT H$ = HEX VALUE (4 DIGIT
  FORMAT)
370 IF DC > 65535 THEN H$ = "ERROR"
: RETURN
375 HV = 4096
: H$ = ""
380 Q% = INT(DC / HV)
: H$ = H$ + MID$(HX$, Q% + 1, 1)
: IF HV > 1 THEN DC = DC - Q% * HV
: HV = HV / 16
: GOTO 380
385 RETURN
390 REM HEX-DEC
395 REM ENTER
: H$ = HEX; EXIT DC = DECIMAL(DC = NEG IF
  ERROR)
400 P = 1
: DC = 0
405 DC = DC + INSTR(HX$, MID$(H$, P, 1)) - 1
: IF P < LEN(H$) THEN DC = DC * 16
: P = P + 1
: GOTO 405
410 RETURN
415 REM END ROUTINE
420 CLS
: PRINT " * * QUIT QBUG * *"
: INPUT "Are you sure (Y/N)"; R$
: IF R$ <> "Y" THEN RETURN
425 GOSUB 435 'CLEAN UP EXISTING BREAKPOINTS
430 MENU
435 REM CLEAR EXISTING BREAKPOINT
440 IF B1(3) <> 0 THEN FOR P = 0 TO 2
: POKE B1(3) + P, B1(P)
: NEXT P
: B1(3) = 0
: RETURN ELSE RETURN
445 REM ACCEPT HEX STRING
450 REM RETURNS HEX IN H$
455 H$ = ""
460 R$ = INKEY$
: IF R$ = "" THEN 460
465 IF INSTR(HX$, R$) THEN H$ = H$ + R$
: PRINT R$;
: GOTO 460
470 IF ASC(R$) = 8 AND LEN(H$) > 0 THEN PRINT
  CHR$(8); " "; CHR$(8);
: H$ = LEFT$(H$, LEN(H$) - 1)
475 IF ASC(R$) = 13 OR ASC(R$) = 31 OR ASC(R$)
  = 30 THEN RETURN
480 GOTO 460
485 REM VIEW USER STACK
490 REM SK = ADDRESS of stack pointer in ML
495 CLS
: PRINT " * * User Stack (at last Breakpoint)
  * *"
: PRINT "Listed most accessible first"
500 N = PEEK(SK)
: IF N = 0 THEN PRINT "-->User stack empty"

```

```

: GOTO 525
505 FOR P = 2 TO N STEP 2
510 LB = PEEK(P + SK)
: HB = PEEK(P + SK + 1)
515 DC = LB
: GOSUB 360
: SK$ = RIGHT$(H$, 2)
520 DC = HB
: GOSUB 360
: SK$ = RIGHT$(H$, 2) + SK$
: PRINT SK$
: NEXT P
525 LINE INPUT "[enter] to Return"; R$
530 RETURN
535 REM HEX-DEC
540 CLS
545 PRINT " * * HEX to Decimal conversion * *"
: PRINT "enter blank to return"
550 PRINT "-->";
: GOSUB 445
: IF H$ = "" THEN RETURN
555 GOSUB 390
: PRINT DC
: GOTO 550
560 REM DECIMAL-HEX
565 CLS
: PRINT " * * Decimal to Hex conversion * *"
: PRINT "enter blank to return"
570 DC = 0
: INPUT "-->"; DC
: IF DC = 0 THEN RETURN
575 GOSUB 360
: PRINT H$
: GOTO 570
580 CALL 30300
: OUT 185, 255
: CL = INP(186)
: OUT 186, CL AND 252
: CL = INP(232)
: CALL 29756
: RETURN
585 REM ML SUPPORT ROUTINE (V1.1)
590 DATA 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  33, 0, 0, 57, 34, 74, 245, 33, 0, 0, 229,
  33, 62, 245
595 DATA 126, 254, 0, 202, 122, 245, 133, 111,
  78, 35, 70, 197, 33, 62, 245
600 DATA 53, 53, 195, 102, 245, 42, 78, 245,
  229, 33, 87, 245, 94, 33, 80, 245, 86,
  213, 35, 70, 35, 78, 35, 86, 35, 94, 42,
  85
605 DATA 245, 241, 201, 245, 34, 85, 245, 33,
  80, 245, 119, 35, 112, 35, 113, 35
610 DATA 114, 35, 115, 35, 209, 123, 50, 87,
  245, 33, 62, 245, 54, 0, 193, 120, 254, 0,
  194, 220, 245, 121
615 DATA 254, 0, 194, 220, 245, 34, 76, 245,
  33, 0, 0, 57, 235, 33, 74, 245, 123, 190,
  202, 210, 245
620 DATA 42, 76, 245, 195, 220, 245, 35, 122,
  190, 200, 42, 76, 245, 195, 220, 245, 126,
  254, 10, 202, 175
625 DATA 245, 198, 2, 119, 133, 111, 113, 35,
  112, 33, 62, 245, 195, 175, 245
630 RESTORE 590
: FOR P! = 62782 TO 62959
: READ PX
: POKE P!, PX
: NEXT
: RETURN

```


ADDRESS NOT UNKNOWN

Software transporting became big business with the introduction of the Tandy 200. Custom-written applications as well as commercial products can be transported to the new machine.

The address table below, contributed by Bill Walters, compares ROM addresses between the Model 100 and the Tandy 200.

The table is based on information received from Tandy. All Tandy 200 ROM calls have the same exit and entry conditions as the Model 100, with differences shown below.

LCD Functions

Name	100	200
LCD	4B44	503C
SETCUR	7440	8D6A
PLOT	744C	8D76
UNPLOT	744D	8D77
POSIT	427C	4F9B
ESCA	4270	4F8F
CRLF	4222	4F3E
HOME	422D	4F3E
CLS	4231	4F4D
SETSYS	4235	4F54
RSTSYS	423A	4F59
LOCK	423F	4F5E
UNLOCK	4244	4F63
CURSON	4249	4F68
CUROFF	424E	4F6D
DELLIN	4253	4F72
INSLIN	4258	4F77
ERAEOL	425D	4F7C
ENTREV	4269	4F88
EXTREV	426E	4F8D

Keyboard

Name	100	200
KYREAD	724H	8B03
CHGET	12CB	12F7
CHSNS	12DB	1404
KEYX	7270	8B31
BRKCHK	7283	8B4D
INLIN	4644	54F6
STFNK	5A7C	6E20

CLRFNK	5A79	6E1D
DSPFNK	42A8	4FC7
STDSPF	42A5	4FC4
ERAFNK	428A	4FA9
FNKSB	5A9E	6E42

Printing

Name	100	200
PRINTR	6D3F	84C9
CHPLPT	1470	1590
PRTTAB	4B55	5A14
PRTLCD	1E5E	2946

RS232 and Modem

Name	100	200
DISC	52BB	61BA
CONN	52D0	61D0
DIAL	532D	622B
RCVX	6D6D	8508
RV232C	6D7E	8519
SENSCQ	6E0B	8608
SENDSC	6E1E	8617
SD232C	6E32	8643
CARDET	6EEF	874A
SNDCOM	6E3A	8629
BAUDST	6E75	86AD
INZCOM	6EA6	86DE
SETSER	17E6	191D
CLSCOM	6ECB	87B5

Cassette

Name	100	200
CTON	14A8	15C0
CTOFF	14AA	15C2
CASIN	14B0	15C8
CSOUT	14C1	15D9
SYNCW	6F46	87D1
SYNCR	6F85	8810
DATAW	6F5B	87E6
DATAR	702A	88B3

RAM files

Name	100	200
MAKTXT	220F	2D7C
CHKDC	5AA9	6E4D
GTXTTB	5AE3	6E8C
KILASC	1FBE	2AB5
INSCHR	6B61	829C
MAKHOL	6B6D	82A8
MASDEL	6B9F	82DA

Miscellaneous

Name	100	200
INITIO	6CD6	841C
IOINIT	6CE0	8439
MENU	5797	67A4
MUSIC	72C5	8BC0
TIME	190F	1A7E
DATE	192F	1A9E
DAY	1962	1AC5

Here's the composition of the communications status string, located at Model 100 location EF3C:

Bit	Description
0-1	Baud Rate: 00 = none 10 = x1 01 = x16 (fix) 11 = x64
2-3	Word length: 00 = 5 10 = 6 01 = 7 11 = 8
4	Parity: 0 = Disable 1 = Enable
5	Parity Set: 0 = Odd 1 = Even
6-7	Stop Bits: 00 = None 10 = 1 01 = 1.5 11 = 2

ALMOST PERFECTION

No computer is without flaws, and unfortunately the Tandy 200 is no exception. Jim Irwin of Comstock, Michigan has studied the Tandy 200 laptop since its introduction, and has noticed a few -- ahem -- undocumented features.

Saving Data

One problem arises when SAVEing data to an existing file. Sometimes the Tandy 200's directory becomes confused and swaps the names of files.

Assume there are two machine language files named TEST.CO and HELLO.CO in one bank's menu. Executing SAVEM TEST.CO from BASIC might save TEST.CO as HELLO.CO, and the existing HELLO.CO's contents could end up as TEST.CO.

This bug only affects machine language and BASIC files, never TEXT, and usually strikes consecutive directory items. KILLing old files before SAVEing updates will eliminate this interesting Tandy 200 feature.

Word Wrap

TELCOM handles word wrap inconsistently. While uploading with a fixed line length, TELCOM should start new lines at a space (hex 20), not within a word. However, lines are occasionally split at exactly the desired length -- sometimes right in the middle of a word.

The clue to this behavioral mystery lies at address 61964, the word-wrap flag. The value of the flag will vary depending on the last use of the Tandy 200's text editor. If the editor was last used in TEXT, then PEEK(61964) = 1, and uploads will have the proper word wrap. If the editor was last invoked from BASIC, where there is no word wrap, then PEEK(61964) = 0, and uploads will consist of broken words.

Before entering TELCOM with the intention of uploading a file, either execute a BASIC program to set the flag or enter TEXT briefly.

Of course, the ability to upload without word wrap might be handy, too. □

PICO BOARD

The term Picocomputer is occasionally used to describe computers of the Model 100 ilk. Just as pico, as a measurement prefix, is smaller than micro, so picocomputers are physically smaller than microcomputers.

The expression is also used by a bulletin-board system dedicated to small portable computers: the PICOSHACK/BBS, located in Shoreview, Minnesota.

Access to PICOSHACK/BBS is free for computer hobbyists and professionals. The system has five public message boards, private user mail, public domain program transfer and how-to features.

The bulletin board also distributes the Info-Mat electronic magazine, which contains weekly product and service reviews, industry news and opinion.

PICOSHACK/BBS can be reached at 300 bits per second (bps) at (612) 482-9100, using eight data bits, one stop bit, no parity

Alan L. Zeichick
Editor

PORTABLE PROGRAM REVIEW

P.O. Box 250
Highland Mill
Camden, ME 04843

FIRST CLASS U.S. POSTAGE PAID CAMDEN, ME PERMIT NO. 5

SUBSCRIBER: PLEASE ADVISE IF ABOVE ADDRESS IS INCORRECT

PORTABLE PROGRAM REVIEW is published monthly and copyright (c) 1986, all rights reserved, by Camden Communications Inc., a Maine corporation with offices at Highland Mill (P.O. Box 250), Camden, Maine 04843, James S. Povec, President. Contents of this publication may not be reproduced in whole or in part unless expressly authorized in writing by the publisher. Tandy and Radio Shack are registered trademarks of Tandy Corporation. U.S. Subscription \$29.97. Please address all subscriptions and inquiries to Nancy Wight, Circulation Director, (207) 236-4365, ISSN: 0883-7295.